THE AMENDMENT

In the Claims

1. (Currently Amended) A method for processing transaction requests from a plurality of clients in a transaction processing system including a communication module, at least one active execution module, and at least one backup execution module, ~~and a communication module~~, wherein at least one execution module includes logic configured to execute transaction programs and further includes transactional state items, the method including the steps of:

receiving a transaction request from a client at the communication module;

routing the transaction request from the communication module to an active execution module;

processing the routed transaction request in the active execution module by executing a transaction program, wherein the step of executing a transaction program ~~processing~~ includes at least one of accessing, reading, creating, updating and removing transactional state items; and

sending a checkpoint message from the active execution module to a corresponding backup execution module if a transactional state item was created, updated or removed.

2. (Original) The method for processing transaction requests of claim 1, wherein a transactional state item may be accessed by a plurality of clients.

3. (Original) The method for processing transaction requests of claim 1, including the step of:

sending an acknowledgement message from the backup execution module to a corresponding active execution module.

4. (Currently Amended) The method for processing transaction requests of claim 1, wherein at least one transactional state item includes the value of a container-managed persistence field of an Enterprise JavaBean entity object and wherein the transaction program includes methods of Enterprise JavaBeans.

5. (Original) The method for processing transaction requests of claim 1, wherein at least one transactional state item is an object of an object-oriented programming language.

6. (Original) The method for processing transaction requests of claim 3, wherein the backup execution module sends the acknowledgement message to the corresponding active execution module after the backup execution module updates the transactional state items corresponding to the checkpoint message.

7. (Original) The method for processing transaction requests of claim 6, including the step of:

sending a response message from the active execution module to the client.

8. (Original) The method for processing transaction requests of claim 1, including the steps of:

updating transactional state items corresponding to the checkpoint message in the backup execution module; and

sending a response message from the backup execution module to the client.

9. (Original) The method for processing transaction requests of claim 1, including the step of:

sending a response message from the active execution module to the client if the routed transaction request does not alter any transactional state item.

10. (Original) The method for processing transaction requests of claim 3, wherein the processing includes:

attempting to obtain a lock for at least one transactional state item corresponding to the routed transaction request; and

if a lock is obtained then performing at least one of accessing, reading, creating, updating and removing transactional state items corresponding to the obtained lock;

wherein the method further includes the step of releasing an obtained lock upon receipt of the acknowledgement message at the active execution module.

11. (Original) The method for processing transaction requests of claim 10, wherein, if at least one shared lock and at least one exclusive lock are obtained then additionally performing the steps of:

releasing at least one obtained shared locks upon completing the processing of the routed transaction request in the active execution module; and

releasing at least one obtained exclusive locks upon receipt of the acknowledgement message at the active execution module.

12. (Currently Amended) The method for processing transaction requests of claim 10, wherein at least one transactional state item includes the value of a container-managed persistence field of

an Enterprise JavaBean entity object and wherein the transaction program includes methods of Enterprise JavaBeans.

13. (Original) The method for processing transaction requests of claim 10, wherein at least one transactional state item is an object of an object-oriented programming language.

14. (Original) The method for processing transaction requests of claim 6, including the step of:

sending at least one of a data update, data remove and data create message from the active execution module.

15. (Original) The method for processing transaction requests of claim 14, including the step of:

at least one of updating, removing and creating a corresponding record in the database module upon receipt of the database update message.

16. (Original) The method for processing transaction requests of claim 15, including the steps of:

sending a completion message from the database management module to the active execution module;

at least one of creating, updating and removing transactional state items in the active execution module corresponding to the received completion message; and

sending a checkpoint message from the active execution module to its corresponding backup execution module.

17. (Currently Amended) The method for processing transaction requests of claim 16, wherein at least one transactional state item includes the value of a container-managed persistence field of an Enterprise JavaBean entity object and wherein the transaction program includes methods of Enterprise JavaBeans.

18. (Original) The method for processing transaction requests of claim 16, wherein at least one transactional state item is an object of an object-oriented programming language.

19. (Original) A method for processing transaction requests in a transaction processing system including the steps of:

starting a processing of a transaction in an execution module;

obtaining a lock to prevent other transactions from accessing at least one transactional state item;

accessing at least one of the at least one transactional state item by the processing of the transaction;

determining if at least one transactional state item accessed by the transaction is not located in the execution module and upon such a determination:

rolling back the processing of the transaction;

retrieving the at least one transactional state item not located in the execution module from the database module and storing it in the execution module; and

restarting the processing of the transaction in the execution module.

20. (Original) The method for processing transaction requests in a transaction processing system of claim 19, further including throwing a programming language exception when rolling back.

21. (Original) The method for processing transaction requests in a transaction processing system of claim 20, wherein the programming language exception includes the identity of the transactional state item not located in the execution module.

22. (Original) The method for processing transaction requests in a transaction processing system of claim 19, wherein the transactional state item not located in the execution module is an Enterprise Java Bean object.

23. (Original) The method for processing transaction requests in a transaction processing system of claim 22, wherein throwing a programming language exception includes sending the primary key of the Enterprise Java Bean object.

24. (Original) The method for processing transaction requests in a transaction processing system of claim 19, wherein the transactional state item not located in the execution module is an object of an object oriented programming language.

25. (Original) A method for processing transaction requests in a transaction processing system including the steps of:

creating, modifying and/or removing a record in a database module;

triggering the sending of a transaction request to a communication module to at create,

modify and/or remove a transactional state item in an execution module;

routing the transaction request from the communication module to an active execution module;

processing the routed transaction request including at least one of accessing, reading, creating, updating and removing transactional state items;

determining if a transactional state item was at least one of created, updated and removed, and upon such a determination sending a checkpoint message from the active execution module to the corresponding backup execution module.

26. (Currently Amended) A method for initializing a transaction processing system ~~including~~ having at least two execution modules, wherein at least one execution module includes logic configured to execute transaction programs and further includes transactional state items, including the steps of:

sending a start operation to an execution module to start the execution module as an active execution module;

sending an operation to a second execution module to cause it to act as a backup execution module to the active execution module;

creating transactional state items in the active execution module by retrieving information corresponding to the transactional state items from a database module;

sending at least one checkpoint message from the active execution module to the second execution module to replicate the created transactional state items.

27. (Currently Amended) The method for initializing a transaction processing system of claim 26, wherein at least one transactional state item includes the value of a container-managed persistence field of an Enterprise JavaBean entity object and wherein the transaction program includes methods of Enterprise JavaBeans.

28. (Original) The method for initializing a transaction processing system of claim 26, wherein at least one transactional state item is an object of an object-oriented programming language.

29. (Original) The method for initializing a transaction processing system of claim 26, wherein the start operation includes information indicating to create at least one transactional state item.

30. (Currently Amended) A method for initializing a transaction processing system having ~~including~~ a plurality of execution modules, wherein at least one execution module includes logic configured to execute transaction programs and further includes transactional state items including the steps of:

sending a start operation to at least one execution module to start the execution module as an active execution module;

sending an operation to at least one other execution module to cause the creation of at least one backup execution module corresponding to the active execution module;

creating transactional state items in the active execution module by retrieving information corresponding to the transactional state items from a database module; and

sending at least one checkpoint message from the active execution module to the at least one other execution module to replicate the created transactional state items.

31. (Currently Amended) A method for failure recovery in a transaction processing system including a plurality of execution modules including the steps of:

creating an active execution module including at least one transaction program and capable of storing transactional state items accessible from multiple clients;

creating at least one backup execution module corresponding to the active execution module;

storing copies of the transactional state items located on the active execution module on the at least one backup execution modules;

detecting a failure in an active execution module; and

promoting one of the at least one backup execution modules corresponding to the failed active execution module to an active execution module.

32. (Currently Amended) The method for failure recovery of claim 31, wherein at least one transactional state item includes the value of a container-managed persistence field of an Enterprise JavaBean entity object and wherein the transaction program includes methods of Enterprise JavaBeans.

33. (Original) The method for failure of claim 31, wherein at least one transactional state item is an object of an object-oriented programming language.

34. (Original) A method for upgrading an execution module in a transaction processing system including a plurality of execution modules including the steps of:

starting a new execution module as a replacement execution module with a new version of transaction programs;

sending a stop message to a second execution module with an old version of transactional programs to stop the second execution module from processing transactions;

causing a communication module to block requests to the second execution module;

sending a start message to the new execution module indicating that it is a replacement execution module for the second execution module;

enabling the replacement execution module to access the transactional state items located in the second execution module;

retrieving at least one of the transactional state items located in the second execution module and creating at least one transactional state item in the new execution module;

causing the communication module to replaces routes to the second execution module with routes to the new execution module.

35. (Original) The method of upgrading of claim 34, wherein the enabling step includes reformatting the transactional state items of the second execution module to the format according to the new execution module.

36. (Currently Amended) The method for upgrading of claim 35, wherein at least one transactional state item includes the value of a container-managed persistence field of an Enterprise JavaBean entity object and wherein the transaction program includes methods of Enterprise JavaBeans.

37. (Original) The method for upgrading of claim 34, wherein at least one transactional state item is an object of an object-oriented programming language.

38. (Original) The method of upgrading of claim 34, wherein the causing of a communication module to block requests occurs before the sending of a stop message.

39. (Original) A method for processing transaction requests in a transaction processing system including at least one active execution module, at least one backup execution module and a communication module, the method includes the steps of:

submitting at least a first and second transaction request from at least one client to the communication module;

modifying the value of a transactional state item with the first transaction, and specifying a read of a value of a transactional state item modified by the first transaction and beginning processing after the first transaction with the second transaction;

processing the first transaction and sending a checkpoint message;

processing the second transaction without sending a response message;

sending the acknowledgement message for the first transaction from the backup execution module to the active execution module; and

sending the response for at least the second transaction from the active the active execution module to the communication module.

40. (Currently Amended) The method for processing transaction requests of claim 39, wherein at least one transactional state item includes the value of a container-managed persistence field of an Enterprise JavaBean entity object and wherein the transaction program includes methods of Enterprise JavaBeans.

41. (Original) The method for processing transaction requests of claim 39, wherein at least one transactional state item is an object of an object-oriented programming language.

42. (Currently Amended) A method for processing transaction requests in a transaction processing system including at least one active execution module, at least one backup execution module and a communication module, the method including the steps of:

submitting at least a first and second transaction request from at least one client to the communication module;

modifying the value of a transactional state item with the first transaction, and specifying a modification of a value of the transactional state item modified by the first transaction and beginning processing after the first transaction with the second transaction;

processing the first transaction and sending a checkpoint message to a backup execution module, the checkpoint message including a first sequence number;

processing the second transaction and sending a checkpoint message to the backup execution module, the checkpoint message including a second sequence number;

processing the checkpoint messages in the backup execution module in an order based on sequence numbers of the checkpoint messages

~~submitting at least two transaction requests from at least one client to the communication module;~~

~~processing at least two transactions and sending separate checkpoint messages for each of the at least two transactions, the checkpoint messages including a sequence number indicating the order of the transactions; and~~

~~processing the checkpoint messages in a backup execution module in an order based on sequence numbers of the checkpoint message~~.

43. (Original) The method for processing transaction requests of claim 42, further including the steps of:

sending an acknowledgement message from the backup execution module to the active execution module upon completion of the processing of the checkpoint message; and

sending a response message from the active execution module to the client.

44. (Original) A transaction processing system including:

logic configured to allow multiple clients to share access to the same transactional state item;

a communication module, the communication module including routing logic configured to receive transaction requests and forward the transaction requests to an active execution module;

a plurality of execution modules, each execution module including:

at least one transaction program and at least one persistent transactional state item, each transaction program including logic configured to process transaction requests; and

access logic configured to access a transactional state item;

and wherein at least one first execution module is configured in an active mode and at least one second execution module is configured in a backup mode, the second execution module containing a copy of at least one transactional state item that is held in the first execution module.

45. (Original) The transaction processing system of claim 44, wherein the at least one transactional state item includes trigger logic configured to execute a second transaction program at a specified time.

46. (Original) The transaction processing system of claim 44, further including transaction management logic configured to restore at least one transactional state item to a pre-transaction value.

47. (Original) The transaction processing system of claim 46, wherein the logic configured to restore transactional state items upon a failure of the transaction program is separate from the transaction program.

48. (Original) The transaction processing system of claim 44, further including transaction management logic configured to restore at least one transactional state item to a pre-transaction value upon a failure of the transaction program.

49. (Original) The transaction processing system of claim 44, in which the execution module includes a plurality of transaction programs.

50. (Original) The transaction processing system of claim 44, in which the execution module includes a plurality of transactional state items.

51. (Original) The transaction processing system of claim 44, wherein at least one first execution module includes logic, separate from any transaction program, configured to send a checkpoint message to at least one second execution module.

52. (Original) The transaction processing system of claim 44, wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module.

53. (Original) The transaction processing system of claim 52, wherein at least one second execution module includes acknowledgement logic configured to receive a checkpoint message and thereupon send an acknowledgement message to at least one first execution module.

54. (Original) The transaction processing system of claim 53, wherein at least one second execution module is configured to update a transactional state item after receiving the checkpoint message and the acknowledgement logic is configured to send the acknowledgement message after the second execution module updates the transactional state item.

55. (Original) The transaction processing system of claim 44, wherein the routing logic is configured to route a plurality of transactions requests to a plurality of execution modules.

56. (Original) The transaction processing system of claim 55, wherein distribution logic is configured to configure the routing logic to route transaction requests based on a partition criteria.

57. (Original) The transaction processing system of claim 56, wherein the partition criteria partitions transactional state items to a plurality of execution modules.

58. (Original) The transaction processing system of claim 55, wherein distribution logic is configured to ensure that an active copy and a backup copy of each transactional state item are stored in separate execution modules.

59. (Original) The transaction processing system of claim 58, wherein the distribution logic is configured to store a transactional state item in one active execution module.

60. (Original) The transaction processing system of claim 59, wherein the distribution logic is configured to store a transactional state item in at least one backup execution module.

61. (Original) The transaction processing system of claim 57, wherein the partition is performed by partition logic, the partition logic including logic configured to identify and partition transactional state items though primary keys.

62. (Original) The transaction processing system of claim 55, wherein a received transaction request is directed to a selected transactional state item; and the routing logic is configured to send a transaction request to the execution module that includes an active copy of the transactional state item.

63. (Original) The transaction processing system of claim 55, wherein the distribution logic is configured to create sets of transactional states that are nearly uniform in size.

64. (Original) The transaction processing system of claim 55, wherein the distribution logic is configured to perform load balancing of the transactional state items among the execution modules.

65. (Original) The transaction processing system of claim 44, further including status change logic configured to change a status of an execution module from a backup mode to an active mode.

66. (Original) The transaction processing system of claim 65, wherein the status change logic is configured to execute the status change based upon unavailability of an active execution module.

67. (Original) The transaction processing system of claim 44, further including at least one client module including logic configured to send transaction requests to the communication module.

68. (Original) The transaction processing system of claim 67, further including an intermediate server;

wherein the client module includes logic configured to send a client request to the intermediate server; and

wherein the intermediate server includes logic configured to send at least one transaction request to the communication module based on at least one of the client requests.

69. (Original) The transaction processing system of claim 44, further including a plurality of hardware modules each of which includes:

at least one of the execution modules, wherein the distribution logic is configured to prevent an active copy and a backup copy of a single transactional state item from being stored in the same hardware module.

70. (Original) The transaction processing system of claim 69, wherein:

each hardware module includes at least one CPU and a memory module; and

the memory module includes at least one execution module.

71. (Original) The transaction processing system of claim 44, wherein at least one execution module includes lock logic configured to associate a lock with at least one transactional state item.

72. (Original) The transaction processing system of claim 71, wherein the lock logic is configured to acquire the lock before access logic accesses an associated transactional state item.

73. (Original) The transaction processing system of claim 71, wherein the lock is exclusive.

74. (Original) The transaction processing system of claim 71, wherein the lock is shared.

75. (Original) The transaction processing system of claim 44, wherein the execution module includes execution logic configured to execute at least a first transaction program, which is configured to maintain at least one item of the transactional state items in a programming-language variable of the transaction program.

76. (Original) The transaction processing system of claim 48, wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module and logic configured to execute at least a first transaction program, which is configured to maintain at least one item of the transactional state items in a programming-language variable of the transaction program.

77. (Original) The transaction processing system of claim 44, wherein at least one transaction program includes a method of a class of an object-oriented programming language.

78. (Original) The transaction processing system of claim 44, wherein:

the at least one transactional state item includes the value of a container-managed persistence field of Enterprise JavaBeans entity object; and

the at least one transaction program includes methods of Enterprise JavaBeans.

79. (Original) The transaction processing of claim 78, wherein the at one transactional state item includes trigger logic including an Enterprise JavaBeans timer configured to trigger the execution of a second transaction program at a specified time.

80. (Original) The transaction processing system of claim 44, wherein:

the at least one transactional state item includes Java Data Objects and the at least one transactional state items including the values of at least one field of the Java Data Objects; and

the at least one transaction program includes Java Data Object methods.

81. (Original) The transaction processing system of claim 44, wherein:

the at least one transactional state item includes objects of an object-oriented language and the at least one transactional state items including the values of fields of the objects of an object-oriented language; and

the at least one transaction program includes methods of an object-oriented language.

82. (Original) The transaction processing system of claim 48 wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module and

the at least one transactional state item includes Java Data Objects and the at least one transactional state items includes the values of at least one field of the Java Data Objects; and

the at least one transaction program includes Java Data Object methods.

83. (Original) The transaction processing system of claim 44, wherein:

the at least one transactional state item includes values of C# fields; and

the at least one transaction program includes C# methods.

84. (Original) The transaction processing system of claim 48 wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module and

the at least one transactional state item includes values of C# fields; and

the at least one transaction program includes C# methods.

85. (Original) The transaction processing system of claim 44, wherein the at least one transactional state item include instances of XML types and the at least one transactional state item includes values of XML elements.

86. (Original) The transaction processing system of claim 48 wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module and at least one transactional state items include instances of XML types and the at least one transactional state item includes values of XML elements.

87. (Original) The transaction processing system of claim 48 wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module and further including:

transaction management logic configured to restore at least one transactional state item to a pre-transaction value upon a failure of the transaction program; and

the at least one transactional state item include Service Data Objects.

88. (Original) The transaction processing system of claim 44, further including:

a database management module configured to receive messages from at least one execution module, and to communicate with at least one database module; and

at least one database module configured to store at least one record.

89. (Original) The transaction processing system of claim 88, wherein at least one execution module includes trigger logic configured to cause the database management module to perform a database update operation after the completion of a transaction program by the execution module.

90. (Original) The transaction processing system of claim 89, wherein the trigger logic modifies at least one transactional state item to indicate the need to update or modify the database module and to indicate the updates needed in the database module.

91. (Original) The transaction processing system of claim 89, wherein

the trigger logic creates at least one transactional state item to indicate the need to update or modify the database module and to indicate the updates needed in the database module.

92. (Original) The transaction processing system of claim 89, wherein at least one transactional state item refers to a second transactional state item which is to be updated in the database module.

93. (Original) The transaction processing system of claim 89, wherein at least one transactional state item refers to a plurality of transactional state items which are to be updated in the database module.

94. (Original) The transaction processing system of claim 89, wherein update logic is configured to process one or more transactional state items associated with multiple database update indications into a single database update operation.

95. (Original) The transaction processing system of claim 89, wherein the trigger logic modifies at least one transactional state item to indicate the need to update or modify the database module and to indicate the updates needed in the database module and update logic is configured to process one or more transactional state items associated with multiple database update indications into a single database update operation.

96. (Original) The transaction processing system of claim 44, further including:

an information processing system.

97. (Original) The transaction processing system of claim 96, wherein

the execution module includes trigger logic to create an update indication to perform an information processing system operation after the completion of the transaction program.

98. (Original) The transaction processing system of claim 97, wherein

the trigger logic creates or modifies at least one transactional state item to indicate the need to perform an information processing system operation.

99. (Original) The transaction processing system of claim 88, further including:

test logic to indicate if a transactional state item corresponding to a transaction request is located in an execution module;

logic responsive to the test logic indicating the transactional state item is not in an execution module and causing:

roll back of the transaction request;

retrieval of the transactional state item from the database module; and

restart of the transaction request.

100. (Original) The transaction processing system of claim 44, further including:

a database program module including logic which provides a database transaction which modifies the database module;

upon receipt of the database transaction database logic sends a transaction request to the communication module to update the transactional state item corresponding to the database transaction.

101. (Original) The transaction processing system of claim 44, further including:

poll logic in the active execution modules configured to poll the database module to determine if the database module changed a transactional state item corresponding to a transactional state item stored on the execution module;

if the logic finds such a transactional state item, the logic causes the execution module to create a local transaction program to update the active execution module and at least one backup execution modules which store the transactional state item.

102. (Original) The transaction processing system of claim 44, wherein:

the active execution module and at least one backup execution modules include logic to store records of the transaction requests they process;

upon the at least one backup execution modules receiving a transaction request logic is configured to determine if a record exists corresponding to the transaction request;

the logic is configured to respond with the record if the record exists on the backup execution module.

103. (Original) The transaction processing system of claim 102, where records include the reply information.

104. (Original) The transaction processing system of claim 44, in which at least one transactional state item represents an auction and at least one transactional state item represents a bid.

105. (Original) The transaction processing system of claim 44, wherein at least one of the transactional state items represents an auction.

106. (Original) The transaction processing system of claim 105, wherein at least one of the transactional state items represents a bid.

107. (Original) The transaction processing system of claim 105, wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module further including:

transaction management logic configured to restore at least one transactional state item to a pre-transaction value upon a failure of the transaction program;

a database management module configured to receive messages from at least one execution module, and to communicate with at least one database module; and

at least one database module configured to store at least one record.

108. (Original) The transaction processing system of claim 44, wherein at least one transactional state item represents a telephone call.

109. (Original) The transaction processing system of claim 108, wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module further including:

transaction management logic configured to restore at least one transactional state item to a pre-transaction value upon a failure of the transaction program.

110. (Original) The transaction processing system of claim 44, wherein at least one transactional state item represents the state of a running application.

111. (Original) The transaction processing system of claim 44, wherein at least one transactional state item represents the state of an execution module of a first application, the first application being different from a second application that includes said at least one transactional state item.

112. (Original) The transaction processing system of claim 44, wherein at least one transactional state item represents at least one radio frequency identification tag.

113. (Original) The transaction processing system of claim 110, wherein the running application includes a transaction program.

114. (Original) The transaction processing system of claim 110, wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module further including:

transaction management logic configured to restore at least one transactional state item to a pre-transaction value upon a failure of the transaction program.

115. (Original) The transaction processing system of claim 44, in which at least one transactional state item represents the state of a hardware module.

116. (Original) The transaction processing system of claim 115, wherein at least one first execution module includes logic configured to send a checkpoint message to at least one second execution module further including:

transaction management logic configured to restore at least one transactional state item to a pre-transaction value upon a failure of the transaction program.

117. (Original) A transaction processing system including:

logic configured to allow multiple clients to share access to the same transactional state item;

a communication module, the communication module including routing logic configured to receive transaction requests and forward the transaction request to an active execution module;

a plurality of execution modules, each execution module including:

at least one transaction program and at least one persistent transactional state item, each transaction program including logic configured to process transaction requests; and

access logic configured to access a transactional state item;

and wherein at least one first execution module is configured in an active mode and includes logic configured to send a checkpoint message to at least one second execution module and at least one second execution module is configured in a backup mode, the second execution module containing a copy of at least one transactional state item that is held in the first execution module;

transaction management logic configured to restore at least one transactional state item to a pre-transaction value upon a failure of the transaction program;

the at least one transactional state item includes the value of a container-managed persistence field of Enterprise JavaBeans entity object; and

the at least one transaction program includes methods of Enterprise JavaBeans.

118. (Original) A transaction processing system including:

logic configured to allow multiple clients to share access to the same transactional state item;

a communication module, the communication module including routing logic configured to receive transaction requests and forward the transaction request to an active execution module;

a plurality of execution modules, each execution module including:

at least one transaction program and at least one persistent transactional state item, each transaction program including logic configured to process transaction requests; and

access logic configured to access a transactional state item;

and wherein at least one first execution module is configured in an active mode and includes logic configured to send a checkpoint message to at least one second execution module and at least one second execution module is configured in a backup mode;

the second execution module containing a copy of at least one transactional state item that is held in the first execution module;

transaction management logic configured to restore at least one transactional state item to a pre-transaction value upon a failure of the transaction program;

the at least one transactional state item includes the values of fields of the objects of an object-oriented language; and

the at least one transaction program includes methods of an object-oriented language.